Compensating Mass Matrix Potential for Constrained Molecular Dynamics

Abhinandan Jain
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109
Abhinandan.Jain@jpl.nasa.gov

Journal of Computational Physics, Jul, '97 (in press)

Subject classification: 65P99, 82A71.

Keywords: Molecular dynamics, algorithms, simulation.

Running head: Compensating Potential for Constrained Molecular Dynamics

Please address all correspondence to:

Abhinandan Jain
Jet Propulsion Laboratory
M/S 198-326
4800 Oak Grove Drive
Pasadena, CA 91109
Abhinandan.Jain@jpl.nasa.gov
Fax: (818) 393-4440

Compensating Mass Matrix Potential for Constrained Molecular Dynamics

Abhinandan Jain
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109
Abhinandan.Jain@jpl.nasa.gov

Abstract

Rigid internal constraints are used in molecular models to speed up molecular dynamics (MD) simulations. It is well recognized that statistical averages from such constrained MD simulations differ by a metric tensor dependent term from similar averages computed using conventional unconstrained MD simulations. Fixman proposed augmenting the standard potential with a compensating term which depends on the metric tensor to nullify the effects of this bias term. However in the absence of tractable algorithms to compute this compensating tensor potential and its gradient its use has been impractical. This paper derives a new algorithm for computing the compensating potential as well as its gradient for tree topology molecular systems. The algorithm is quite straightforward and is an extension of the spatial operators based $O(\mathcal{N})$ algorithm that has been recently proposed for constrained dynamics. Indeed, the compensating potential is closely related and computed from the articulated body inertia quantities available from this $O(\mathcal{N})$ algorithm.

1 Introduction

Rigid internal constraints are often used in molecular models to eliminate high frequency modes and to enable the use of large numerical integration time steps necessary for speeding up molecular dynamics (MD) simulations [1, 2]. There has been considerable debate regarding the relationship between the statistical averages obtained from such constrained MD simulations and those obtained from conventional unconstrained Cartesian model MD simulations [3–7]. In particular, Fixman [3] pointed out that ensemble averages obtained from MD simulation using the constrained and unconstrained models will differ due to the presence of a metric tensor dependent term in the partition function for constrained molecular models. and this has been verified in simulations. Fixman proposed the augmentation of the standard potential by a compensating metric tensor potential in constrained MD simulations to compensate for the effects of the bias term. Several researchers [8–11] have verified the efficacy of this method for simple molecular systems. The prohibitive complexity of computing the metric tensor potential and its gradient has been a major hurdle on the use of these compensating potentials more generally in constrained MD simulations.

This paper analyzes the structure of the compensating potential and its gradient and develops substantially simpler expressions for them for tree topology molecular models. These expressions are used to derive computational algorithms for use in constrained MD simulations. The algorithms are straightforward extensions of the recently proposed spatial operators based $O(\mathcal{N})$ algorithm for constrained MD simulations [2]. Indeed, the compensating potential and its gradi-

ent are readily computable from the articulated body inertia quantities available from this $O(\mathcal{N})$ algorithm.

2 Ensemble Averages in Constrained Dynamics Simulations

The partition function $\mathcal{Z}(\mathcal{T})$ for an n degree of freedom unconstrained Cartesian molecular model is given by the expression

$$\mathcal{Z}(\mathcal{T}) = \frac{1}{h^n} \int_{-\infty}^{\infty} \cdots\cdots \int_{-\infty}^{\infty} \int_{-\alpha_n}^{\gamma_n} \cdots \int_{-\alpha_1}^{\gamma_1} \exp\left[-\left(\frac{1}{2} p^* \boldsymbol{\mathcal{M}}^{-1} p + \mathcal{V} \right) / k\mathcal{T} \right] \times dp_1 \cdots dp_n dq_1 \cdots dq_n \tag{2.1}$$

where $\mathcal{T}$ denotes the temperature, the q_i and p_i are the configuration and momentum coordinates, $\mathcal{V}$ is the standard potential energy, $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n \times n}$ denotes the system **mass matrix** (or *metric tensor*). The system kinetic energy is given by $\frac{1}{2} p^* \boldsymbol{\mathcal{M}}^{-1} p$, and the limits of integration α_i and γ_i are determined by the geometry of the problem [12]. For conventional unconstrained Cartesian dynamics, the mass matrix $\boldsymbol{\mathcal{M}}$ is constant (and diagonal) and does not depend upon the system configuration. As a consequence, the ensemble average of a function $f(q)$ is given by

$$\langle f(q) \rangle = \frac{2\pi k \mathcal{T}^{n/2}}{h^n} \int_{-\alpha_n}^{\gamma_n} \cdots \int_{-\alpha_1}^{\gamma_1} f(q) \exp\left[-\mathcal{V}/k\mathcal{T} \right] \times dq_1 \cdots dq_n / \mathcal{Z}(\mathcal{T}) \tag{2.2}$$

where the momentum variables have been integrated over and eliminated.

In constrained dynamics models, the molecular system is modeled as a collection of rigid **clusters** coupled together by articulable **hinges** [2]. For such constrained dynamics models, the mass matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{\theta}) \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ is a function of the internal configuration coordinates vector $\boldsymbol{\theta}$ and $\mathcal{N}$ the number of degrees of freedom for the constrained model. The kinetic energy is now given by the expression

$$K.E. = \frac{1}{2} \boldsymbol{\beta}^* \boldsymbol{\mathcal{M}}(\boldsymbol{\theta}) \boldsymbol{\beta} = \frac{1}{2} p^* \boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta}) p \tag{2.3}$$

where $\boldsymbol{\beta}$ denotes the internal velocity coordinates vector, and where the conjugate momenta vector p is given by the expression

$$p = \boldsymbol{\mathcal{M}}(\boldsymbol{\theta}) \boldsymbol{\beta} \tag{2.4}$$

The partition function $\mathcal{Z}(\mathcal{T})$ for the system is given by the expression

$$\mathcal{Z}'(\mathcal{T}) = \frac{1}{h^{\mathcal{N}}} \int_{-\infty}^{\infty} \cdots\cdots \int_{-\infty}^{\infty} \int_{-\alpha_{\mathcal{N}}}^{\gamma_{\mathcal{N}}} \cdots \int_{-\alpha_1}^{\gamma_1} \exp\left[-\left(\frac{1}{2} p^* \boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta}) p + \mathcal{V} \right) / k\mathcal{T} \right] \times dp'_1 \cdots dp'_{\mathcal{N}} d\boldsymbol{\theta}_1 \cdots d\boldsymbol{\theta}_{\mathcal{N}} \tag{2.5}$$

Using a diagonalizing transformation on the momentum coordinates of the form

$$p' = \boldsymbol{\mathcal{M}}^{-\frac{1}{2}}(\boldsymbol{\theta}) p \tag{2.6}$$

we can integrate Eq. (2.5) over the new momentum coordinates to get the following expression for the partition function:

$$\mathcal{Z}(\mathcal{T}) = \frac{2\pi k \mathcal{T}^{\mathcal{N}/2}}{h^{\mathcal{N}}} \int_{-\alpha_{\mathcal{N}}}^{\gamma_{\mathcal{N}}} \cdots \int_{-\alpha_1}^{\gamma_1} \det\left\{ \boldsymbol{\mathcal{M}}^{\frac{1}{2}}(\boldsymbol{\theta}) \right\} \exp\left[-\mathcal{V}/k\mathcal{T} \right] \times d\boldsymbol{\theta}_1 \cdots d\boldsymbol{\theta}_{\mathcal{N}} \tag{2.7}$$

This implies that the ensemble average of a function $f(\boldsymbol{\theta})$ over the configuration space defined by the internal coordinates is given by the following expression:

$$\langle f(\boldsymbol{\theta})\rangle' = \frac{2\pi k\mathcal{T}^{\mathcal{N}/2}}{h^{\mathcal{N}}} \int_{-\alpha_{\mathcal{N}}}^{\gamma_{\mathcal{N}}} \cdots \int_{-\alpha_1}^{\gamma_1} \det\left\{\boldsymbol{\mathcal{M}}^{\frac{1}{2}}(\boldsymbol{\theta})\right\} f(\boldsymbol{\theta}) \exp\left[-\mathcal{V}/k\mathcal{T}\right] \times d\boldsymbol{\theta}_1 \cdots d\boldsymbol{\theta}_{\mathcal{N}}/\mathcal{Z}'(\mathcal{T}) \tag{2.8}$$

In contrast with the unconstrained Cartesian dynamics expression in Eq. (2.2), the integrand in the partition function integral in Eq. (2.5) involves the determinant of the mass matrix. This additional term introduces a bias in statistical averages computed using constrained dynamics models versus those obtained using Cartesian models [3].

A method for bridging this gap in statistical estimates was proposed by Fixman [3]. He suggested replacing the potential function $\mathcal{V}(\boldsymbol{\theta})$ by the modified potential function $\mathcal{V}'(\boldsymbol{\theta})$

$$\mathcal{V}'(\boldsymbol{\theta}) \stackrel{\triangle}{=} \mathcal{V}(\boldsymbol{\theta}) + \mathcal{V}_c(\boldsymbol{\theta}), \quad \text{where} \quad \mathcal{V}_c(\boldsymbol{\theta}) \stackrel{\triangle}{=} \frac{1}{2}\ln\det\{\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\} \tag{2.9}$$

in constrained MD simulations. It is easy to see that replacing $\mathcal{V}(theta)$ in Eq. (2.7) and Eq. (2.8) by $\mathcal{V}'(\boldsymbol{\theta})$ eliminates the metric tensor term from the partition functions and the expression for the ensemble average. The extra potential term $\mathcal{V}_c(\boldsymbol{\theta})$, is referred to as the **compensating mass matrix potential** or the *metric-tensor potential* [7] since it effectively compensates for the mass matrix determinant term in Eq. (2.8).

With the use of $\mathcal{V}'(\boldsymbol{\theta})$, the remaining differences between statistical averages computed using constrained and unconstrained MD simulations are due to the coarser sampling of the conformational space during the averaging process in constrained MD simulations. The appropriate use of constraints is important for ensuring that the loss in fidelity is within acceptable limits for the simulation experiment at hand. The use of the compensating mass matrix potential helps ensure that at least the systematic bias term from the metric tensor does not contribute to the averaging errors.

The implication of using $\mathcal{V}'(\boldsymbol{\theta})$ is that now its gradient must be used for computing the forces during constrained MD simulations. The overall hinge torque vector T' is defined as the gradient of $\mathcal{V}'(\boldsymbol{\theta})$ and is given by:

$$T' = \nabla_{\boldsymbol{\theta}}\mathcal{V}'(\boldsymbol{\theta}) = T + T_c, \quad \text{where} \quad T \stackrel{\triangle}{=} \nabla_{\boldsymbol{\theta}}\mathcal{V}(\boldsymbol{\theta}) \quad \text{and} \quad T_c \stackrel{\triangle}{=} \nabla_{\boldsymbol{\theta}}\mathcal{V}_c(\boldsymbol{\theta}) \tag{2.10}$$

T_c represents the compensating hinge torque arising from the compensating mass-matrix potential $\mathcal{V}_c(\boldsymbol{\theta})$ and must be used in addition to the standard torque term T during constrained MD simulations. Its k^{th} element, $T_c(k)$, for the k^{th} hinge is given by

$$T_c(k) = \frac{\partial\mathcal{V}_c(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}(k)} = \frac{1}{2}\frac{\partial\ln\det\{\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\}}{\partial\boldsymbol{\theta}(k)} \tag{2.11}$$

The need for the use of the compensating potential has been verified via computer simulations for several small systems [7,8,10,11,13,14]. There is general consensus that the compensating potential is less important for rigid constraints on the bond stretching degrees of freedom while they are a significant factor for rigid constraints involving bond angles. It was found [7,8] that the use of the compensating potential in constrained MD simulations for n-butane other systems effectively

bridged the gap in the number of dihedral transitions using constrained and unconstrained MD simulations. Go and Scheraga [5, 6] examined the relative merits of Cartesian dynamics models and constrained dynamics models and concluded that the former was the more correct model for molecular dynamics simulations. However references [8, 9] argued that the constrained MD simulations produce similar statistical estimates as the Cartesian case if the compensating potential is included.

Despite the accepted importance of using the compensating potential during constrained dynamics simulations, it is rarely used in practice. The primary factor has been the lack of a tractable method for computing the compensating torque, T_c, for all but simple molecular systems [13]. A clever method for computing $\mathcal{V}_c$ was proposed by Fixman [4]. However, this method is also limited to moderately sized molecular systems, and does not include a procedure for computing the compensating torque T_c.

In the following sections we take a closer look at the expression for T_c in Eq. (2.11) and derive simpler expressions for it using techniques from the **spatial operator algebra** [2]. These expressions lead to a simple method for computing this compensating torque for arbitrary tree-topology molecular systems. This method is an extension of the spatial operators based algorithm for constrained dynamics simulations [2]. This algorithm provides a highly efficient $O(\mathcal{N})$ recursive method for solving the equations of motion for constrained systems without using iterative procedures such as in the SHAKE algorithm [15]. The equations of motion for the system are solved exactly and the complexity of the algorithm is $O(\mathcal{N})$, i.e. the computational cost grows only linearly with the number of un-constrained degrees of freedom in the system. The algorithm – which has been implemented as the **NEIMO** (Newton–Euler Inverse Mass Operator method) software package [2, 16, 17] is based upon closed-form spatial operator expressions for the factorization and inversion of the mass matrix.

Expression for the Compensating Torque $T_c(k)$

In general, if $g(X)$ is a scalar function of a matrix $X \in \mathbb{R}^{m\times n}$, then its derivative with respect to a variable y is given by

$$\frac{\partial g(X)}{\partial y} = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial g(X)}{\partial X(i,j)} \frac{\partial X(i,j)}{\partial (y)} = \text{Trace}\left\{ \frac{\partial g}{\partial X}^{*} \frac{\partial X}{\partial y} \right\} \tag{2.12}$$

where $\frac{\partial g(X)}{\partial X(i,j)}$ and $\frac{\partial X(i,j)}{\partial y}$ are $m \times n$ matrices whose elements are defined as:

$$\frac{\partial g}{\partial X}(i,j) \triangleq \frac{\partial g(X)}{\partial X(i,j)}, \quad \text{and} \quad \frac{\partial X}{\partial y}(i,j) \triangleq \frac{\partial X(i,j)}{\partial y}$$

For the scalar function $g(X) \triangleq \ln\det\{X\}$, it is a well established fact [18] that

$$\frac{\partial g(X)}{\partial X} = \frac{\partial \ln\det\{X\}}{\partial X} = \{X^{*}\}^{-1} \tag{2.13}$$

Using Eq. (2.13) and Eq. (2.12) in Eq. (2.11) leads to the following expression for $T_c(k)$:

$$T_c(k) = \frac{1}{2}\,\text{Trace}\left\{ \boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta}) \frac{\partial \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}(k)} \right\} = \frac{1}{2}\,\text{Trace}\left\{ \boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta}) \boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}(k)}(\boldsymbol{\theta}) \right\} \tag{2.14}$$

We have used the notational shorthand $\boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}(k)}(\boldsymbol{\theta})$ in place of $\frac{\partial \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}(k)}$ in Eq. (2.14). In the following sections we use spatial operator expressions for $\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})$ to further simplify Eq. (2.14).

3 Spatial Operator form of Mass Matrix

Now we briefly review the derivation of the spatial operator equations of motion and refer the reader to reference [2] for more detailed discussion on the notation and the concepts. For notational simplicity we limit our initial discussion to an n-cluster system with *serial chain* structure and single degree of freedom rotational hinges between clusters. The number of degrees of freedom for this system is $\mathcal{N} = n + 5$. Later we discuss the steps involved in extending the derivations and algorithms to general tree-topology systems with multiple degree of freedom hinges.

As shown in reference [2], the Newton-Euler recursive equations of motion for the whole system have the form:

$$\left\{\begin{array}{l} \qquad V(n+1) = 0, \quad \alpha(n+1) = 0 \\ \textbf{for } \boldsymbol{k} \;=\; \boldsymbol{n \cdots 1} \\ \quad V(k) \;=\; \boldsymbol{\phi}^*(k+1,k)V(k+1) + \boldsymbol{H}^*(k)\dot{\boldsymbol{\theta}}(k) \\ \quad \alpha(k) \;=\; \boldsymbol{\phi}^*(k+1,k)\alpha(k+1) + \boldsymbol{H}^*(k)\ddot{\boldsymbol{\theta}}(k) + a(k) \\ \textbf{end loop} \end{array}\right. \tag{3.1}$$

$$\left\{\begin{array}{l} \qquad f(0) = 0 \\ \textbf{for } \boldsymbol{k} \;=\; \boldsymbol{1 \cdots n} \\ \quad f(k) \;=\; \boldsymbol{\phi}(k,k-1)f(k-1) + \boldsymbol{M}(k)\alpha(k) + b(k) + \hat{f}_c(k) \\ \quad T(k) \;=\; \boldsymbol{H}(k)f(k) \\ \textbf{end loop} \end{array}\right.$$

The introduction of **spatial operators** allows the expression of the equations of motion in the following more concise form:

$$\begin{array}{rcl} V &=& \boldsymbol{\phi}^*\boldsymbol{H}^*\dot{\boldsymbol{\theta}} \\ \alpha &=& \boldsymbol{\phi}^*(\boldsymbol{H}^*\ddot{\boldsymbol{\theta}} + a) \\ f &=& \boldsymbol{\phi}(\boldsymbol{M}\alpha + b + \hat{f}_c) = \boldsymbol{\phi}\boldsymbol{M}\boldsymbol{\phi}^*\boldsymbol{H}^*\ddot{\boldsymbol{\theta}} + \boldsymbol{\phi}(\boldsymbol{M}\boldsymbol{\phi}^*a + b + \hat{f}_c) \\ T &=& \boldsymbol{H}f = \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{M}\boldsymbol{\phi}^*\boldsymbol{H}^*\ddot{\boldsymbol{\theta}} + \boldsymbol{H}\boldsymbol{\phi}(\boldsymbol{M}\boldsymbol{\phi}^*a + b + \hat{f}_c) \end{array} \tag{3.2}$$

In particular, the equations of motion have the form

$$T = \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{\mathcal{C}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \tag{3.3}$$

where

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\theta}) \;\triangleq\; \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{M}\boldsymbol{\phi}^*\boldsymbol{H}^* \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} \tag{3.4a}$$

$$\boldsymbol{\mathcal{C}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \;\triangleq\; \boldsymbol{H}\boldsymbol{\phi}(\boldsymbol{M}\boldsymbol{\phi}^*a + b + \hat{f}_c) \in \mathbb{R}^{\mathcal{N}} \tag{3.4b}$$

Here, $\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})$ is the **mass matrix** of the serial chain and $\boldsymbol{\mathcal{C}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the vector of Coriolis, centrifugal, gyroscopic and Cartesian forces. Note that $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{C}}$ are nonlinear functions of $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$. The factorization in Eq. (3.4) of the mass matrix $\boldsymbol{\mathcal{M}}$ is referred to as the **Newton–Euler Operator**

Factorization [19] because it is equivalent to the recursive Newton–Euler inverse dynamics algorithm in Eq. (3.1).

The solution of the equations of motion in Eq. (3.3) for the accelerations vector $\ddot{\boldsymbol{\theta}}$ is used by the numerical integrator to propagate the state of the system during molecular dynamics simulations. However, Eq. (3.3) represents only a conceptual statement of the dynamics problem since $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{C}}$ are not explicitly available. The conventional approach for computing the accelerations $\ddot{\boldsymbol{\theta}}$ consists of first computing both $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{C}}$, and solving the linear matrix equation for the vector $\ddot{\boldsymbol{\theta}}$. In general, $\boldsymbol{\mathcal{M}}$ is fully populated and as a result, the computational cost of solving the equations of motion using this method grows cubically with the number of degrees of freedom in the system, i.e., this method is of $O(\mathcal{N}^3)$ computational complexity. The computational advantage of larger integration time-steps using constrained dynamics [20] can be lost due to the large computational costs for large molecules from the $O(\mathcal{N}^3)$ dependency.

In the next section we review an alternative recursive algorithm for computing the vector of generalized accelerations $\ddot{\boldsymbol{\theta}}$ without having to explicitly compute the mass matrix [2]. The complexity of this method is only $O(\mathcal{N})$, i.e., its computational cost grows only linearly with the number of degrees of freedom in the model.

Innovations Factorization of the Mass Matrix

The $O(\mathcal{N})$ spatial algebra algorithm for solving the equations of motion in Eq. (3.3) described in [2] depends on the following key results that give explicit analytical operator expressions for the square factorization and inversion of the mass matrix.

Lemma 3.1

$$\boldsymbol{\mathcal{M}} = [I + \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{K}]\boldsymbol{D}[I + \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{K}]^* \tag{3.5a}$$

$$[I + \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{K}]^{-1} = [I - \boldsymbol{H}\boldsymbol{\psi}\boldsymbol{K}] \tag{3.5b}$$

$$\boldsymbol{\mathcal{M}}^{-1} = [I - \boldsymbol{H}\boldsymbol{\psi}\boldsymbol{K}]^*\boldsymbol{D}^{-1}[I - \boldsymbol{H}\boldsymbol{\psi}\boldsymbol{K}] \tag{3.5c}$$

Proof: *See reference [2].* ∎

The new square factorization described in Eq. (3.5a) is also referred to as the **Innovations Operator Factorization** of the mass matrix and is an alternative to the factorization in Eq. (3.4). The factor $[I + \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{K}] \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is square, block lower triangular and nonsingular, while $\boldsymbol{D}$ is a block diagonal matrix. This factorization provides a closed form operator expression for the block LDL^* decomposition of $\boldsymbol{\mathcal{M}}$. The following lemma gives the closed form operator expression for the inverse of the factor $[I + \boldsymbol{H}\boldsymbol{\phi}\boldsymbol{K}]$. Once again, the factor $[I - \boldsymbol{H}\boldsymbol{\psi}\boldsymbol{K}]$ is square, block lower triangular and nonsingular and so Eq. (3.5c) provides a closed form expression for the block LDL^* decomposition of $\boldsymbol{\mathcal{M}}^{-1}$. The spatial operators $\boldsymbol{\phi}$, $\boldsymbol{K}$ and $\boldsymbol{D}$ embedded in these factorizations are based on spatially recursive filtering and smoothing algorithms [19, 21, 22]. The following Riccati equation for the articulated body inertia $\boldsymbol{P}$ is a key element of these filtering and smoothing algorithms.

Algorithm 3.1

The articulated body inertia quantities $\boldsymbol{P}(.)$, $\boldsymbol{D}(.)$, $\boldsymbol{G}(.)$, $\boldsymbol{K}(.)$, $\boldsymbol{\tau}(.)$, $\overline{\boldsymbol{\tau}}(.)$, $\boldsymbol{P}^+(.)$ *and* $\boldsymbol{\psi}(.,.)$ *are*

computed by the following recursive procedure:

$$\left\{\begin{array}{rcl} \boldsymbol{P}^+(0) = 0 & & \\ \textbf{for } \boldsymbol{k} \;=\; \boldsymbol{1\cdots\mathcal{N}} & & \\ \boldsymbol{P}(k) & = & \boldsymbol{\phi}(k,k-1)\boldsymbol{P}^+(k-1)\boldsymbol{\phi}^*(k,k-1)+\boldsymbol{M}(k) \\ \boldsymbol{D}(k) & = & \boldsymbol{H}(k)\boldsymbol{P}(k)\boldsymbol{H}^*(k) \\ \boldsymbol{G}(k) & = & \boldsymbol{P}(k)\boldsymbol{H}^*(k)\boldsymbol{D}^{-1}(k) \\ \boldsymbol{K}(k+1,k) & = & \boldsymbol{\phi}(k+1,k)\boldsymbol{G}(k) \\ \boldsymbol{\tau}(k) & = & \boldsymbol{G}(k)\boldsymbol{H}(k) \\ \overline{\boldsymbol{\tau}}(k) & = & \boldsymbol{I}-\boldsymbol{\tau}(k) \\ \boldsymbol{P}^+(k) & = & \overline{\boldsymbol{\tau}}(k)\boldsymbol{P}(k) \\ \boldsymbol{\psi}(k+1,k) & = & \boldsymbol{\phi}(k+1,k)\overline{\boldsymbol{\tau}}(k) \\ \textbf{end loop} & & \end{array}\right. \tag{3.6}$$

Algorithm 3.1 is the by now classical [19, 23] Riccati equation of Kalman filtering. Its solution $\boldsymbol{P}(k)$ is the articulated body inertia [19,24] of the part of the system outboard of hinge k. The operator $\boldsymbol{P}$ is a block-diagonal $6n\times 6n$ matrix with its k^{th} diagonal element being $\boldsymbol{P}(k)\in\mathbb{R}^{6\times 6}$. Define also

$$\begin{array}{rcl} \boldsymbol{D} & = & \boldsymbol{HPH}^* \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} \\ \boldsymbol{G} & = & \boldsymbol{PH}^*\boldsymbol{D}^{-1} \in \mathbb{R}^{6n\times\mathcal{N}} \\ \boldsymbol{K} & = & \boldsymbol{\mathcal{E}_\phi G} \in \mathbb{R}^{6n\times n} \\ \overline{\boldsymbol{\tau}} & = & \boldsymbol{I}-\boldsymbol{GH} \in \mathbb{R}^{6n\times 6n} \\ \boldsymbol{\mathcal{E}_\psi} & = & \boldsymbol{\mathcal{E}_\phi}\overline{\boldsymbol{\tau}} \in \mathbb{R}^{6n\times 6n} \\ \boldsymbol{\psi} & = & (\boldsymbol{I}-\boldsymbol{\mathcal{E}_\psi})^{-1} \in \mathbb{R}^{6n\times 6n} \end{array} \tag{3.7}$$

The operators $\boldsymbol{D}$, $\boldsymbol{G}$ and $\overline{\boldsymbol{\tau}}$ are all block diagonal. The operators $\boldsymbol{K}$ and $\boldsymbol{\mathcal{E}_\psi}$ are not block-diagonal, but their only nonzero block elements are $\boldsymbol{K}(k,k-1)$'s and $\boldsymbol{\psi}(k,k-1)$'s respectively along the first sub-diagonal. The following lemma describes the operator expression for the generalized accelerations $\ddot{\boldsymbol{\theta}}$ in terms of the hinge forces T and Cartesian spatial forces $\hat{f}_c$.

Lemma 3.2

$$\ddot{\boldsymbol{\theta}} = [I-\boldsymbol{H\psi K}]^*\boldsymbol{D}^{-1}\Big[T-\boldsymbol{H\psi}\{\boldsymbol{K}T+\boldsymbol{P}a+b+\hat{f}_c\}\Big]-\boldsymbol{K}^*\boldsymbol{\psi}^*a \tag{3.8}$$

Proof: *See reference [2].* ∎

$O(\mathcal{N})$ Algorithm for Solving the Equations of Motion

The recursive implementation of Eq. (3.8) leads to the following $O(\mathcal{N})$ computational algorithm

for the accelerations, $\ddot{\boldsymbol{\theta}}$:

$$\left\{\begin{array}{rcl} & z^+(0) = 0 & \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{1 \cdots n} \\ z(k) & = & \boldsymbol{\phi}(k, k-1) z^+(k-1) + \boldsymbol{P}(k) a(k) + b(k) + \hat{f}_c(k) \\ \boldsymbol{\epsilon}(k) & = & T(k) - \boldsymbol{H}(k) z(k) \\ \boldsymbol{\nu}(k) & = & \boldsymbol{D}^{-1}(k) \boldsymbol{\epsilon}(k) \\ z^+(k) & = & z(k) + \boldsymbol{G}(k) \boldsymbol{\epsilon}(k) \\ \textbf{end loop} & & \end{array}\right. \tag{3.9}$$

$$\left\{\begin{array}{rcl} & \alpha(n+1) = 0 & \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{n \cdots 1} \\ \alpha^+(k) & = & \boldsymbol{\phi}^*(k+1, k) \alpha(k+1) \\ \ddot{\boldsymbol{\theta}}(k) & = & \boldsymbol{\nu}(k) - \boldsymbol{G}^*(k) \alpha^+(k) \\ \alpha(k) & = & \alpha^+(k) + \boldsymbol{H}^*(k) \ddot{\boldsymbol{\theta}}(k) + a(k) \\ \textbf{end loop} & & \end{array}\right. \tag{3.10}$$

This algorithm does not require either the explicit computation of the mass matrix $\boldsymbol{\mathcal{M}}$, nor the numerical solution of the matrix equation Eq. (3.3). The steps in the above algorithm can be summarized as follows:

1. The first step is a recursion from the base to the tip to compute the orientation, location and spatial velocities, $V(k)$, and the Coriolis and gyroscopic terms $a(k)$ and $b(k)$ for each of the clusters using the first base-to-tip recursion in Eq. (3.1).
2. Next follows a recursion from the tip towards the base as defined by Eq. (3.6) to compute the $\boldsymbol{P}(k)$'s etc.
3. The recursion in Eq. (3.9) from the tip to the base is used next to compute the residual forces $z(k)$ etc. This recursion can be combined with the tip to base recursion in the previous step to obtain a single tip to base recursion sequence.
4. Finally, the base to tip recursion described by Eq. (3.10) computes the $\ddot{\boldsymbol{\theta}}(k)$ accelerations for all the clusters.

The computational cost of this algorithm depends only linearly on the number of clusters. The structure of this algorithm closely resembles those found in Kalman filtering and smoothing theory [21, 25].

4 Compensating Mass Matrix Torque $T_c(i)$

Using Eq. (2.14) as a starting point, we now develop an expression for the compensating torque $T_c(i)$ that is simple to compute. While we have already seen an expression for the mass matrix

inverse, $\boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta})$, we need an expression for the derivative of the mass matrix with respect to hinge coordinates.

Spatial Operator Expression for $\boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}_i}$
The following lemma gives this desired expression for the sensitivity matrix.

Lemma 4.1

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}_i} = \boldsymbol{H}\boldsymbol{\phi}\left[\mathbb{H}_\delta^i\boldsymbol{\phi}\boldsymbol{M} - \boldsymbol{M}\boldsymbol{\phi}^*\mathbb{H}_\delta^i\right]\boldsymbol{\phi}^*\boldsymbol{H}^* \tag{4.11}$$

Proof: *See reference [26].* ∎

The matrix $\mathbb{H}_\delta^i$ a new quantity in this result. $\mathbb{H}_\delta^i$ is the $6n \times 6n$ matrix whose elements are all zero, except for a single 6×6 block $\mathbb{H}(i)$ at the i^{th} location on the diagonal. The index i corresponds to the joint-angle $\boldsymbol{\theta}_i$ with respect to which the sensitivity $\boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}_i}$ is being taken. The non-zero block-diagonal element $\mathbb{H}(i) \in \mathbb{R}^{6\times 6}$ is obtained as follows from the hinge rotational axis unit vector $h(i)$:

$$\mathbb{H}(i) = \begin{pmatrix} \tilde{h}(i) & 0 \\ 0 & \tilde{h}(i) \end{pmatrix} \tag{4.12}$$

The notation $\tilde{v}$ above denotes the 3×3 *cross-product tensor matrix* associated with a 3-vector $v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ and is defined as

$$\tilde{v} \triangleq \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

The formula in Eq. (4.11) is closed-form, in the sense that it contains an explicit analytical expression for the mass matrix sensitivity in terms of the operators $\boldsymbol{\phi}$, $\boldsymbol{M}$, and $\boldsymbol{H}$ appearing in the mass matrix itself. That the formula is closed-form is of extreme importance, because it implies that the mass matrix derivatives can be easily computed using operations and spatially recursive algorithms similar to those used to compute the mass matrix itself.

Spatial Operator Expressions for $T_c(i)$
The following lemma uses Lemma 4.11 to develop a new expression for $T_c(i)$.

Lemma 4.2

$$T_c(i) = \text{Trace}\left\{\boldsymbol{P}\boldsymbol{\Omega}\mathbb{H}_\delta^i\right\}, \quad \textit{where} \quad \boldsymbol{\Omega} \triangleq \boldsymbol{\psi}^*\boldsymbol{H}^*\boldsymbol{D}^{-1}\boldsymbol{H}\boldsymbol{\psi} \in \mathbb{R}^{6n\times 6n} \tag{4.13}$$

Proof: *Two spatial operator identities that we need for the proof are given by the following equations. Their derivation can be found in the appendix of reference [27].*

$$[\boldsymbol{I} - \boldsymbol{H}\boldsymbol{\psi}\boldsymbol{K}]\boldsymbol{H}\boldsymbol{\phi} = \boldsymbol{H}\boldsymbol{\psi} \tag{4.14a}$$

$$\boldsymbol{\phi}\boldsymbol{M}\boldsymbol{\Omega} = (\boldsymbol{\phi} - \boldsymbol{\psi}) + \boldsymbol{P}\boldsymbol{\Omega} \tag{4.14b}$$

The derivation of Eq. (4.13) goes as follows:

$$
\begin{aligned}
T_c(i) &\overset{2.11}{=} \frac{1}{2}\,\mathrm{Trace}\left\{\boldsymbol{\mathcal{M}}^{-1}(\boldsymbol{\theta})\boldsymbol{\mathcal{M}}_{\boldsymbol{\theta}(k)}(\boldsymbol{\theta})\right\} \\
&\overset{3.5c,4.11}{=} \frac{1}{2}\,\mathrm{Trace}\left\{[I-\boldsymbol{H\psi K}]^*\boldsymbol{D}^{-1}[I-\boldsymbol{H\psi K}]\boldsymbol{H\phi}\left[\mathbb{H}_\delta^i\boldsymbol{\phi M}-\boldsymbol{M\phi}^*\mathbb{H}_\delta^i\right]\boldsymbol{\phi}^*\boldsymbol{H}^*\right\} \\
&\overset{4.14a}{=} \mathrm{Trace}\left\{[I-\boldsymbol{H\psi K}]^*\boldsymbol{D}^{-1}\boldsymbol{H\psi}\mathbb{H}_\delta^i\boldsymbol{\phi M\phi}^*\boldsymbol{H}^*\right\} \\
&= \mathrm{Trace}\left\{\boldsymbol{\phi}^*\boldsymbol{H}^*[I-\boldsymbol{H\psi K}]^*\boldsymbol{D}^{-1}\boldsymbol{H\psi}\mathbb{H}_\delta^i\boldsymbol{\phi M}\right\} \\
&\overset{4.14a}{=} \mathrm{Trace}\left\{\boldsymbol{\psi}^*\boldsymbol{H}^*\boldsymbol{D}^{-1}\boldsymbol{H\psi}\mathbb{H}_\delta^i\boldsymbol{\phi M}\right\} \\
&\overset{4.13}{=} \mathrm{Trace}\left\{\boldsymbol{\Omega}\mathbb{H}_\delta^i\boldsymbol{\phi M}\right\} \\
&= \mathrm{Trace}\left\{\boldsymbol{\phi M\Omega}\mathbb{H}_\delta^i\right\} \\
&\overset{4.14b}{=} \mathrm{Trace}\left\{(\boldsymbol{\phi}-\boldsymbol{\psi}+\boldsymbol{P\Omega})\mathbb{H}_\delta^i\right\} \\
&= \mathrm{Trace}\left\{\boldsymbol{P\Omega}\mathbb{H}_\delta^i\right\}
\end{aligned}
$$

In the above steps, we have used the fact that $\mathrm{Trace}\{AB\} = \mathrm{Trace}\{BA\}$. *Also, the last step used the fact that* $\mathrm{Trace}\{(\boldsymbol{\phi}-\boldsymbol{\psi})\mathbb{H}_\delta^i\} = 0$. *This is true because* $(\boldsymbol{\phi}-\boldsymbol{\psi})$ *is strictly lower triangular and* $\mathbb{H}_\delta^i$ *is block diagonal.* ∎

It has been shown in reference [19] that $\boldsymbol{\Omega}$ can be decomposed as follows:

$$\boldsymbol{\Omega} = \boldsymbol{\Upsilon} + \tilde{\boldsymbol{\psi}}^*\boldsymbol{\Upsilon} + \boldsymbol{\Upsilon}\tilde{\boldsymbol{\psi}} \tag{4.15}$$

where $\tilde{\boldsymbol{\psi}} \overset{\triangle}{=} \boldsymbol{\psi} - I$, and the diagonal elements $\boldsymbol{\Upsilon}(k,k) \in \mathbb{R}^{6\times 6}$ of the block diagonal matrix $\boldsymbol{\Upsilon} \in \mathbb{R}^{6n\times 6n}$ are defined via the following recursion:

$$
\left\{\begin{array}{l}
\qquad\quad \boldsymbol{\Upsilon}(n+1) = 0 \\
\textbf{for } \boldsymbol{k} \;=\; \boldsymbol{n\cdots 1} \\
\quad \boldsymbol{\Upsilon}(k) \;=\; \boldsymbol{\psi}^*(k+1,k)\boldsymbol{\Upsilon}(k+1)\boldsymbol{\psi}(k+1,k) + \boldsymbol{H}^*(k)\boldsymbol{D}^{-1}(k)\boldsymbol{H}(k) \\
\textbf{end loop}
\end{array}\right. \tag{4.16}
$$

This allows us to further simplify the expression for $T_c(i)$ as described in the following lemma.

Lemma 4.3

$$T_c(i) = \mathrm{Trace}\{\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i)\mathbb{H}(i)\} \tag{4.17}$$

Proof: *Using Eq. (4.15) in Eq. (4.13) it follows that*

$$
\begin{aligned}
T_c(i) &= \mathrm{Trace}\left\{\boldsymbol{P\Omega}\mathbb{H}_\delta^i\right\} = \mathrm{Trace}\left\{\boldsymbol{P}(\boldsymbol{\Upsilon}+\tilde{\boldsymbol{\psi}}^*\boldsymbol{\Upsilon}+\boldsymbol{\Upsilon}\tilde{\boldsymbol{\psi}})\mathbb{H}_\delta^i\right\} \\
&= \mathrm{Trace}\left\{\boldsymbol{P\Upsilon}\mathbb{H}_\delta^i\right\} = \mathrm{Trace}\{\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i)\mathbb{H}(i)\}
\end{aligned}
$$

Here we used the fact that $\mathrm{Trace}\left\{\boldsymbol{P\Upsilon}\tilde{\boldsymbol{\psi}}\mathbb{H}_\delta^i\right\} = 0$ *because* $\tilde{\boldsymbol{\psi}}$ *is strictly lower triangular.* ∎

A final simplification step for $T_c(i)$ is described in the following lemma.

Lemma 4.4 *Let the partitioned form of the* 6 *matrix* $\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i)$ *be given by:*

$$\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i) = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$$

where $Q_{ij} \in \mathbb{R}^{3\times 3}$. *Then*

$$T_c(i) = -h^*(i)\mathcal{F}[Q_{11} + Q_{22}] \tag{4.18}$$

where the mapping $\mathcal{F}[\cdot] : \mathbb{R}^{3\times 3} \rightarrow \mathbb{R}^3$ *is defined via the relation:*

$$v = \mathcal{F}[A] \quad \textit{if and only if} \quad \tilde{v} = A - A^*$$

Proof: *We have*

$$\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i)\mathbb{H}(i) = \begin{pmatrix} Q_{11}\tilde{h}(i) & Q_{22}\tilde{h}(i) \\ Q_{21}\tilde{h}(i) & Q_{22}\tilde{h}(i) \end{pmatrix} \tag{4.19}$$

Therefore

$$\text{Trace}\left\{\boldsymbol{P}(i)\boldsymbol{\Upsilon}(i)\mathbb{H}(i)\right\} = \text{Trace}\left\{(Q_{11} + Q_{22})\tilde{h}(i)\right\}$$

Using the easily established identity that $\text{Trace}\{A\tilde{v}\} = -v^*\mathcal{F}[A]$ *for an arbitrary matrix* $A \in \mathbb{R}^{3\times 3}$ *and vector* $v \in \mathbb{R}^3$ *in Eq. (4.19) establishes the result.* ∎

Notice that this expression for $T_c(i)$ is vastly simpler compared with the original expression involving the mass matrix inverse and its sensitivity. Furthermore, the expression involves articulated body inertia quantities many of whom are available from the computational steps for solving the equations of motion. It is easy to verify that the the compensating torque vector for the six base cluster degrees of freedom is zero.

$O(\mathcal{N})$ Constrained Dynamics Algorithm with Compensating Potential

The overall algorithm for solving the equations of motion with the compensating mass matrix potential is defined by the following steps:

1. Carry out the first base-to-tip part of the recursion in Eq. (3.1) to compute the $V(k)$, $a(k)$ and $b(k)$ terms for all the clusters.

2. Carry out the tip-to-base recursion in Eq. (3.6) to compute all the articulated body inertia quantities such as $\boldsymbol{P}$, $\boldsymbol{D}$ etc.

3. Carry out the base-to-tip recursion in Eq. (4.16) to compute $\boldsymbol{\Upsilon}(k)$ for all the links. Also compute the compensating torque $T_c(k)$ for each link simultaneously using Eq. (4.18) and $T'(k)$ using Eq. (2.10).

4. Carry out the recursions in Eq. (3.9) and Eq. (3.10) to solve for the $\ddot{\boldsymbol{\theta}}(k)$ hinge accelerations with $T(k)$ replaced with $T'(k)$.

The only significant change from the constrained dynamics algorithm in [2] is the additional Step 3 for the compensating torque. This additional step is also of $O(\mathcal{N})$ computational complexity, and hence the overall computational cost of the constrained dynamics algorithm remains $O(\mathcal{N})$. Moreover this method adds only marginal computational cost since it makes use of the articulated body inertia quantities available from the computations for the regular solution for the internal coordinate accelerations. This algorithm makes possible the easy incorporation of the compensating potential into constrained molecular dynamics simulations involving tree-topology molecular systems. This algorithm has been implemented as a part of the **NEIMO** software package.

Extensions

Reference [28] describes an alternative method for computing the $\boldsymbol{\Upsilon}(k)$ terms using dual articulated body inertias instead of Eq. (4.16). This method offers advantages for parallel implementation since these computations can be done concurrently with Step 2 instead of sequentially following it.

The derivations and algorithmic descriptions in the previous sections have focused upon serial-chain molecules with one degree of freedom rotational hinges. This was done for notational simplicity. The extension of the algorithm to tree-topology molecules is identical to the extension discussed in reference [2]. In this case the system has multiple tips and a single designated base cluster. To summarize, all the tip-to-base and base-to-tip recursions are replaced by tips-to-base and base-to-tips recursions. At each hinge with branches, the recursions proceed through "scatter" and "gather" steps. To handle multiple degree of freedom hinges, it is simply a matter of recognizing that such hinges can be modeled as a sequence of single degree of freedom hinges inter-connected by pseudo-clusters of zero mass and inertia. With these modification, the algorithm described above extends to general tree-topology molecular systems.

Spatial Operator Expression for $\mathcal{V}_c$

While the explicit knowledge of the compensating potential $\mathcal{V}_c(\boldsymbol{\theta})$ is not required during dynamics simulations, the nature and sensitivity of its dependence on the configuration coordinates can be used to study its effect on the statistical averages obtained from constrained dynamics simulations. The following lemma gives a closed form expression for the compensating potential $\mathcal{V}_c$.

Lemma 4.5

$$\mathcal{V}_c(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} \ln \det \{\boldsymbol{D}(i)\} \tag{4.20}$$

Proof: *From the Innovations factorization in Eq. (3.5a), it follows that*

$$\det\{\boldsymbol{\mathcal{M}}\} = \det\{I + \boldsymbol{H\phi K}\}^2 \det\{\boldsymbol{D}\}$$

However, since $[I + \boldsymbol{H\phi K}]$ is lower triangular with identity matrix blocks along its diagonal, its determinant is 1. Hence

$$\det\{\boldsymbol{\mathcal{M}}\} = \det\{\boldsymbol{D}\}$$

The result of the lemma follows simply by taking the logarithm of the above identity. ∎

The expression in Eq. (4.20) allows the easy computation of $\mathcal{V}_c$ by making use of Algorithm 3.1 for computing the required $\boldsymbol{D}(k)$ quantities.

5 Conclusions

Internal rigid constraints together with internal coordinates are often used to speed up molecular dynamics computations. However, it has been recognized that the use of such constrained dynamics introduces systematic biases into the computation of statistical averages, and a compensating mass matrix potential is required to offset these biases. The lack of a tractable method for computing these compensating terms has rendered their use impractical to date.

This paper derives analytical expressions and algorithms for including the compensating mass matrix potential and its gradient into constrained MD simulations for general tree-topology molecular systems. The algorithm is an extension of the previously described $O(\mathcal{N})$ algorithms for internal coordinate molecular dynamics simulations. The computational complexity of the new algorithm remains $O(\mathcal{N})$. Extension of this compensating potential algorithm to closed-topology molecular systems is the subject of ongoing research.

6 Acknowledgments

The author would like to thank Dr. N. Vaidehi for reviewing the paper and for many helpful suggestions. The research described in this paper has been performed partially at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration and with support from NSF grant ACS 92-17368.

References

[1] A. Mazur and R. Abagyan, "New Methodology for Computer–Aided Modeling of Biomolecular Structure and Dynamics: 1. Non–Cyclic Structures," *J. of Biomolecular Structure& Dynamics*, vol. 6, no. 4, pp. 815–832, 1989.

[2] A. Jain, N. Vaidehi, and G. Rodriguez, "A Fast Recursive Algorithm for Molecular Dynamics Simulations," *Journal of Computational Physics*, vol. 106, pp. 258–268, June 1993.

[3] M. Fixman, "Classical Statistical Mechanics of Constraints: A Theorem and Application to Polymers," *Proc. Nat. Acad. Sci.*, vol. 71, pp. 3050–3053, Aug. 1974.

[4] M. Fixman, "Simulation of Polymer Dynamics. I. General Theory," *J. Chem. Phys.*, vol. 69, pp. 1527–1537, Aug. 1978.

[5] N. Go and H. Scheraga, "Analysis of the Contribution of Internal Vibrations to the Statistical Weights of Equilibrium Conformations of Macromolecules," *J. Chem. Phys.*, vol. 51, pp. 4751–4767, Dec. 1969.

[6] N. Go and H. Scheraga, "On the Use fo Classical Statistical Mechanics in the Treatment of Polymer Chain Conformation," *Macromolecules*, vol. 9, pp. 535–542, July 1976.

[7] W. van Gunsteren and M. Karplus, "Effects of Constraints on the Dynamics of Macromolecules," *Macromolecules*, vol. 15, pp. 1528–1544, Nov. 1982.

[8] M. Pear and J. Weiner, "Brownian dynamics study of a polymer cahin of linked rigid bodies," *J. Chem. Phys.*, vol. 71, pp. 212–224, July 1979.

[9] E. Helfand, "Flexible vs rigid constraints in statistical mechanics," *J. Chem. Phys.*, vol. 71, pp. 5000–5007, Dec. 1979.

[10] D. Chandler and B. Berne, "Comment on the role of constraints on the conformational structure of n-butane in lid solventseiqu," *J. Chem. Phys.*, vol. 71, pp. 5386–5387, Dec. 1979.

[11] W. van Gunsteren, "Constrained Dynamics of Flexible of Molecules," *Molecular Physics*, vol. 40, no. 4, pp. 1015–1019, 1980.

[12] A. Munster, *Statistical Thermodynamics*. Springer-Verlag, 1969.

[13] H. Berendsen and W. van Gunsteren, *The Physics of Superionic Conductors and Electrode Materials*, pp. 221–240. Plenum Press, 1980.

[14] M. Allen and D. Tildesley, *Computer Simulation of Liquids*. Clarendon Press, 1987.

[15] J. Mcammon and S. Harvey, *Dynamics of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1987.

[16] A. M. Mathiowetz, A. Jain, N. Karasawa, and W. A. Goddard III, "Protein Simulations using Techniques Suitable for Very Large Systems: the Cell Multipole Method for Nonbond Interactions and the Newton-Euler Inverse Mass Operator Method for Internal Coordinate Dynamics," *Proteins: Structure, Function, and Genetics*, vol. 20, pp. 227–247, 1994.

[17] N. Vaidehi, A. Jain, and W. Goddard, "Constatnt Temperature Constrained Molecular Dynamics: The Newton-Euler Inverse Mass Operator Method," *J. Phys. Chem.*, vol. 100, no. 25, pp. 10508–10517, 1996.

[18] A. Graham, *Kronecker products and matrix calculus: with applications*. Halsted Press, 1981.

[19] G. Rodriguez, K. Kreutz-Delgado, and A. Jain, "A Spatial Operator Algebra for Manipulator Modeling and Control," *The International Journal of Robotics Research*, vol. 10, pp. 371–381, Aug. 1991.

[20] A. Mazur, V. Dorofeev, and R. Abagyan, "Derivation and Testing of Explicit Equations of Motion for Polymers Described by Internal Coordinates," *Journal of Computational Physics*, vol. 92, pp. 261–272, Feb. 1991.

[21] G. Rodriguez, "Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 624–639, Dec. 1987.

[22] G. Rodriguez and K. Kreutz-Delgado, "Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 65–76, Feb. 1992.

[23] T. Kailath, "The Innovations Approach to Detection and Estimation Theory," *Proceedings of the IEEE*, vol. 58, pp. 680–695, Mar. 1970.

[24] R. Featherstone, "The Calculation of Robot Dynamics using Articulated-Body Inertias," *The International Journal of Robotics Research*, vol. 2, pp. 13–30, Spring 1983.

[25] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Prentice-Hall Inc., 1979.

[26] A. Jain and G. Rodriguez, "Diagonalized Lagrangian Robot Dynamics," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 571–584, Aug. 1995.

[27] A. Jain and G. Rodriguez, "An Analysis of the Kinematics and Dynamics of Underactuated Manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 411–422, Aug. 1993.

[28] A. Jain and G. Rodriguez, "Base-Invariant Symmetric Dynamics of Free–Flying Space Manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 585–597, Aug. 1995.